STRUCTURED SPARSITY WITH REPARAMETERIZED BATCH NORMALIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Group sparsity is a practical approach to accelerating neural network inference, while reparameterization provides a promising mechanism for inducing sparsity. Existing reparameterization-based approaches to group sparsity generally apply reparameterization in filter space. However, this is not the only possible location for inducing sparsity. In this work, we find empirically that applying reparameterization in the Batch Normalization (BN) space achieves better accuracy–sparsity tradeoffs and identifies higher-quality sparse architectures. Experiments further show that BN-space reparameterization is substantially more robust to the BN stabilization parameter ϵ than filter-space reparameterization. To understand these observations, we examine the optimization dynamics and find that BN suppresses the task sensitivity of filter-space gates, whereas BN-space gates remain coupled to the task loss, allowing task feedback to guide sparsification consistently.

1 INTRODUCTION

Sparsity in deep learning models has received considerable attention in recent years as an effective way to reduce model complexity and computational cost (Han et al., 2015; 2016; Zhu & Gupta, 2017; Frankle & Carbin, 2019; Hoefler et al., 2021). Existing sparsification methods can be broadly divided into *unstructured sparsity* and *structured sparsity* (Wen et al., 2016; Hoefler et al., 2021). Unstructured sparsity prunes individual parameters and can achieve high compression ratios, but the resulting irregular sparsity is often difficult to translate into practical acceleration on standard hardware (Han et al., 2015; 2016; Zhu & Gupta, 2017; Molchanov et al., 2017; Louizos et al., 2018; Gale et al., 2019; Li et al., 2017). Structured sparsity instead prunes entire filters, channels, attention heads, or other parameter groups, producing smaller dense networks that can be directly executed by existing deep-learning libraries and hardware (Wen et al., 2016; Li et al., 2017; He et al., 2017; Luo et al., 2017; Liu et al., 2017; Gordon et al., 2018; Michel et al., 2019). This makes structured sparsity particularly attractive for practical model compression and acceleration (Wen et al., 2016; Li et al., 2017; Liu et al., 2017).

In parallel, reparameterization-based sparse optimization has emerged as a promising approach to inducing sparsity (Hoff, 2017; Poon & Peyré, 2021; Ziyin & Wang, 2023). Instead of directly optimizing a non-smooth or nonconvex sparse regularizer, these methods represent an effective parameter through multiple trainable factors and optimize a smooth objective over the reparameterized variables (Hoff, 2017; Poon & Peyré, 2021; 2023; Dai et al., 2021; Ziyin & Wang, 2023; Kolb et al., 2026). Although this idea has been explored and shown to be effective, most of this line of work has focused on unstructured sparsity (Ziyin & Wang, 2023; Li et al., 2021; Kolb et al., 2025b; Xu et al., 2026). Recently, D-Gating (Kolb et al., 2025a) extended multiplicative reparameterization to structured sparsity by associating parameter groups with reparameterized gates.

However, group sparsity can be induced through different components of a network (Wen et al., 2016; Liu et al., 2017; Huang & Wang, 2018; Gordon et al., 2018; You et al., 2019), and the choice of where to apply sparsity-inducing reparameterization has received relatively little attention. Existing approaches such as D-Gating place multiplicative reparameterization in convolutional filter space (Kolb et al., 2025a). In networks with Batch Normalization, however, scaling a filter before BN also scales its activation statistics, so the resulting change can be largely canceled by normalization (Liu et al., 2017; Ye et al., 2018). Consequently, shrinking a filter-space gate does not necessarily lead to a comparable reduction in the normalized channel output (Liu et al., 2017; Ye et al., 2018),

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

and the gate magnitude may no longer faithfully reflect the functional strength of the channel (Ye et al., 2018). This observation motivates us to examine how the placement of sparsity-inducing reparameterization affects networks with batch normalization.

In this paper, we propose BN-Gating, which applies reparameterization in the Batch Normalization (BN) space, and compare it with filter-space reparameterization. Experiments on CIFAR-10, CIFAR-100, and ImageNet-100 show that BN-Gating consistently retains higher accuracy at comparable sparsity levels, with larger gains in the high-sparsity regime. For example, the accuracy gap reaches 1.23% near 87% sparsity on CIFAR-10 and 2.23% near 78% sparsity on CIFAR-100. To further understand where this advantage comes from, we compare the sparse architectures identified by the two methods at similar sparsity levels and retrain them from scratch under the same training setup. BN-Gating consistently yields higher retraining accuracy, indicating that its advantage is not only due to better inherited weights, but also to identifying better sparse architectures.

We further investigate why the gate location matters. For a filter-space gate placed before BN, scaling the filter also scales its BN statistics, so the effect is largely canceled by normalization and the task loss remains weakly sensitive to the gate over a broad range of values. In contrast, a BN-space gate directly scales the normalized feature and therefore remains visible to the task objective throughout sparsification. This difference leads to distinct sparsification dynamics: D-Gating receives little task feedback when the gate is large and undergoes a delayed, concentrated channel selection process, whereas BN-Gating is continuously shaped by task feedback and forms its sparse structure more gradually. We validate this mechanism in two controlled experiments. First, we place a gate either before BN or on the BN scale of the same channel and vary it while keeping all other parameters fixed. Second, we vary the BN stabilization parameter ϵ, which directly controls the degree to which filter-space scaling is masked by normalization.

Finally, we examine how multiplicative reparameterization itself affects sparsity formation in BN space. Our depth ablation shows that larger gating depths induce sparsity more aggressively, while smaller depths behave more conservatively. We further find that multiplicative reparameterization promotes a polarized distribution of BN scaling factors, with sparse channels concentrated near zero and active channels clearly separated from them. This separation makes channel selection less ambiguous and reduces the dependence on post-pruning recovery. We further compare BN-Gating with representative group-sparsity baselines and find that it achieves a better sparsity–accuracy tradeoff, even compared with baselines after post-pruning fine-tuning.

2 Related Work

Early studies connected multiplicative reparameterization with explicit sparse regularization. Hoff (2017) showed that Hadamard product parameterizations can transform a class of ℓ_q-regularized problems into equivalent objectives with quadratic penalties on the factorized variables. This idea was later developed into smooth reformulations of non-smooth sparse optimization. Poon & Peyré (2021) proposed smooth overparameterized formulations for sparse penalties such as the Lasso and group Lasso, and further extended this framework to more general structured regularizers (Poon & Peyré, 2023). From a related representation-cost perspective, Dai et al. (2021) showed that factorized parameterizations combined with simple ℓ_2 regularization can induce ℓ_p-type and group-sparse regularization on the effective parameters.

Beyond such explicit reformulations, multiplicative parameterization can also induce sparsity implicitly through its optimization dynamics. Studies of diagonal linear networks have shown that gradient-based optimization under multiplicative parameterization can favor sparse solutions, with the induced bias depending on factors such as initialization and factorization depth (Li et al., 2021). This perspective has also been extended from element-wise sparsity to group sparsity, where grouped diagonal parameterizations were shown to favor group-sparse solutions (Li et al., 2023). These results suggest that multiplicative reparameterization affects sparsity not only through the explicit regularizer, but also through the optimization dynamics induced by the parameterization.

More recent work has developed reparameterization into a practical mechanism for sparse optimization in neural networks. SPRED (Ziyin & Wang, 2023) showed that ℓ_1-regularized objectives can be reformulated into smooth factorized objectives and optimized with standard gradient-based methods. Smoothing the Edges (Kolb et al., 2026) generalized this idea through Hadamard over-

parameterization to broader classes of unstructured and structured sparse penalties. Subsequent work investigated how the form of the reparameterization affects sparsity formation. Deep Weight Factorization (Kolb et al., 2025b) showed that increasing the factorization depth induces stronger nonconvex sparsity-promoting effects, while ReWA (Xu et al., 2026) studied the interaction between reparameterization, weight decay, and learning rate and introduced adaptive learning rates for sparse optimization. Recently, D-Gating (Kolb et al., 2025a) extended multiplicative reparameterization from individual parameters to parameter groups by associating convolutional filters and other structures with factorized gates, enabling differentiable structured sparsification.

Structured sparsity has been extensively studied as a practical approach to network compression and acceleration (Li et al., 2017; He et al., 2017; Luo et al., 2017; Gordon et al., 2018; Molchanov et al., 2019; He et al., 2019). Structured Sparsity Learning (Wen et al., 2016) applies group regularization to filters, channels, filter shapes, and layers to learn hardware-friendly sparse structures. Sparse Structure Selection (Huang & Wang, 2018) introduces learnable scaling factors for different network structures and sparsifies them during end-to-end training, while Gate Decorator (You et al., 2019) associates channel-wise gates with convolutional filters and uses their effect on the task objective to guide global pruning. Other approaches further investigate how channel importance should be represented or learned instead of relying solely on filter magnitude (He et al., 2020; Molchanov et al., 2019; He et al., 2019).

A particularly related line of work uses Batch Normalization parameters for channel selection. Network Slimming (Liu et al., 2017) induces channel sparsity by applying sparsity regularization directly to the BN scaling factors. Ye et al. (2018) further questioned the smaller-norm-less-informative assumption commonly used in filter pruning, showing that filter magnitudes can become ambiguous when followed by BN, since rescaling a filter can leave the normalized activation largely unchanged. These observations motivate BN scaling factors as a more direct representation of channel strength.

The two lines of work above leave a complementary question open. Reparameterization-based methods mainly study how a sparse parameterization should be constructed and optimized, while structured-pruning methods study which variables should represent channel importance. In this work, we connect these perspectives by asking where multiplicative reparameterization should be applied in networks with Batch Normalization. Specifically, we compare reparameterization in convolutional filter space with reparameterization of the corresponding BN scaling factors, and propose BN-Gating.

3 COMPARISON OF BN-SPACE AND FILTER-SPACE REPARAMETERIZATION

This section compares BN-space and filter-space reparameterization from both algorithmic and empirical perspectives. We first explain the algorithm of D-Gating and BN-Gating, and then compare their sparsity–accuracy tradeoffs and the quality of the sparse architectures they identify.

3.1 D-GATING: EXISTING FILTER-SPACE REPARAMETERIZATION

We first review D-Gating, an existing approach that induces group sparsity through multiplicative reparameterization in filter space. A depth-D parameterization consists of D learnable factors in total: one filter factor A and $D-1$ channel-wise gating factors. The effective filter is $W = A \odot s$, where $s = \prod_{i=1}^{D-1} S_i$ acts as a channel-wise gate.

The reparameterized filter factors are jointly optimized with the task objective and a quadratic penalty on both the filter factor A and the gating factors:

$$\min_{A,\{S_i\}} \mathcal{L}\left(A \odot \prod_{i=1}^{D-1} S_i\right) + \frac{\lambda}{2D}\left(\|A\|_2^2 + \sum_{i=1}^{D-1} \|S_i\|_2^2\right), \tag{1}$$

where λ controls the strength of sparsity induction. The optimization procedure is summarized in Algorithm 1. After training, channels with sufficiently small effective filter norms are pruned.

Algorithm 1 D-Gating

Require: Filter factor A, channel-wise gating factors $\{S_i\}_{i=1}^{D-1}$, learning rate η_t, regularization strength λ, pruning threshold τ

Define the channel-wise inner product $\left[\langle U, V\rangle_{\text{ch}}\right]_c = \sum_{k,h,w} U_{c,k,h,w} V_{c,k,h,w}$

for $t = 0, \ldots, T-1$ **do**

$W^{(t)} = A^{(t)} \odot \prod_{i=1}^{D-1} S_i^{(t)}$

$G^{(t)} = \nabla_{W^{(t)}} \mathcal{L}$

$$A^{(t+1)} = A^{(t)} - \eta_t \left[G^{(t)} \odot \prod_{i=1}^{D-1} S_i^{(t)} + \frac{\lambda}{D} A^{(t)} \right]$$

$$S_i^{(t+1)} = S_i^{(t)} - \eta_t \left[\left\langle G^{(t)}, A^{(t)} \right\rangle_{\text{ch}} \odot \prod_{\substack{j=1 \\ j \neq i}}^{D-1} S_j^{(t)} + \frac{\lambda}{D} S_i^{(t)} \right], \quad i = 1, \ldots, D-1$$

end for

$W^{(T)} = A^{(T)} \odot \prod_{i=1}^{D-1} S_i^{(T)}$

Set $W_c^{(T)} = 0$ for each channel c with $\|W_c^{(T)}\|_2 < \tau$

3.2 BN-Gating: Proposed BN-Space Reparameterization

BN-Gating applies the same multiplicative reparameterization to the BN scaling factor instead of the convolutional filter. For channel c, let γ_c denote its BN scale and parameterize it as

$$\gamma_c = \prod_{i=1}^{D} S_{i,c}.$$

The corresponding optimization objective is

$$\min_{A, \{S_i\}} \mathcal{L}\left(A, \prod_{i=1}^{D} S_i\right) + \frac{\lambda}{2D} \sum_{i=1}^{D} \|S_i\|_2^2. \tag{2}$$

Thus, D-Gating places the reparameterized gate on the convolutional filter before BN, whereas BN-Gating directly reparameterizes the BN scale after normalization. The detailed optimization procedure is given in Algorithm 2.

Pruning interpretation. BN-Gating and D-Gating use the same interpretation of a pruned conv–BN channel: its output no longer depends on the input, so the convolution producing that channel can be skipped while its constant output is retained. In ResNet, this applies to the residual branch without removing the corresponding shortcut channel. Thus, the two methods are compared using the same channel-counting principle. Details are given in Appendix D.The idealized FLOPs of the compact channel configurations selected by BN-Gating are reported in Appendix C.

3.3 Comparison of Sparsity–Accuracy Tradeoffs

To evaluate the effectiveness of BN-Gating and D-Gating, we compare their sparsity–accuracy tradeoffs across multiple datasets and network architectures. Specifically, we conduct experiments on CIFAR-10, CIFAR-100, and ImageNet-100 using both VGG and ResNet architectures. By varying the sparsity strength, we obtain models at different sparsity levels and compare the corresponding test accuracy. Table 1 summarizes the results.

Across these settings, BN-Gating consistently achieves higher accuracy at comparable sparsity levels, with the advantage becoming more pronounced at high sparsity. On CIFAR-10, the accuracy gain increases from 0.35 pp at around 75–77% sparsity to 1.23 pp near 87% sparsity. On CIFAR-100, the advantage of BN-Gating grows with sparsity, reaching 0.59 pp near 68% sparsity and 2.23 pp near 78% sparsity. On ImageNet-100, BN-Gating also improves accuracy by 0.58 pp at a comparable sparsity level. Results are reported as mean $\pm$ standard deviation over multiple seeds.* denotes linearly interpolated D-Gating results when no nearby measured point is available.

Algorithm 2 BN-Gating

Require: BN reparameterization factors $\{S_i\}_{i=1}^{D}$, task gradient $G^{(t)} = \nabla_{\gamma^{(t)}}\mathcal{L}$, learning rate η_t, regularization strength λ, pruning threshold τ

1: **for** $t \in [T]$ **do**

2: $\gamma^{(t)} = \prod_{i=1}^{D} S_i^{(t)}$

3: $S_i^{(t+1)} = S_i^{(t)} - \eta_t \left[G^{(t)} \prod_{\substack{j=1 \\ j \neq i}}^{D} S_j^{(t)} + \frac{\lambda}{D} S_i^{(t)} \right], \quad i = 1, \ldots, D$

4: **end for**

5: Prune the corresponding channel if $\left|\gamma^{(T)}\right| < \tau$

Table 1: Comparison of sparsity–accuracy tradeoffs between BN-Gating and D-Gating. * denotes values obtained by linear interpolation.

Dataset	BN-Gating		D-Gating		Δ Acc.
	Sparsity (%)	Acc. (%)	Sparsity (%)	Acc. (%)	(pp)
CIFAR-10	77.23 ± 0.46	93.11 ± 0.16	74.66 ± 1.92	92.76 ± 0.38	$+0.35$
	83.35 ± 0.30	93.13 ± 0.21	81.99 ± 1.53	92.17 ± 0.17	$+0.96$
	86.94 ± 0.22	92.23 ± 0.14	87.76 ± 1.27	91.00 ± 0.53	$+1.23$
CIFAR-100	30.74 ± 0.32	75.62 ± 0.15	30.74^*	75.65^*	-0.03
	47.98 ± 0.50	74.86 ± 0.23	44.23 ± 3.06	74.61 ± 0.98	$+0.25$
	68.19 ± 0.27	72.74 ± 0.21	68.54 ± 2.93	72.15 ± 1.28	$+0.59$
	78.20 ± 0.10	70.96 ± 0.29	78.20^*	68.73^*	$+2.23$
ImageNet-100	55.64 ± 0.73	83.45 ± 0.13	57.83 ± 0.23	82.87 ± 0.11	$+0.58$

Table 2: Accuracy of sparse architectures retrained from scratch. Results are reported as mean $\pm$ standard deviation over three random seeds.

Dataset	Method	Sparsity (%)	Accuracy (%)
CIFAR-10	BN-Gating	83.12	91.68 ± 0.09
CIFAR-10	D-Gating	83.03	91.13 ± 0.13
CIFAR-100	BN-Gating	73.83	68.04 ± 0.50
CIFAR-100	D-Gating	73.83	67.82 ± 0.26

3.4 Quality of Selected Sparse Architectures

We select pairs of models with similar sparsity levels and retrain the corresponding architectures from scratch to further evaluate the quality of the sparse structures. Specifically, the pruned architectures are randomly initialized and trained under the same training configuration, without inheriting weights from the original models. This comparison isolates the quality of the selected network structures from the effect of inherited parameters. The retraining results are reported in Table 2.

At nearly identical sparsity levels, the architecture selected by BN-Gating achieves higher accuracy after retraining from scratch. This result suggests that BN-Gating identifies a higher-quality sparse architecture, rather than merely benefiting from better inherited weights.

4 Optimization Dynamics of BN-Space and Filter-Space Reparameterization

This section investigates why the location of reparameterization leads to different sparsification behavior. We first analyze the task-loss sensitivity of filter-space and BN-space gates, showing that Batch Normalization weakens task-driven updates to filter-space gates, whereas BN-space gates continue to receive direct task signals. We then validate this mechanism through controlled gate-response and BN-ϵ experiments, and finally examine how the resulting difference in task feedback shapes gate evolution and channel-selection dynamics during training.

4.1 Task-Loss Feedback under Different Gate Placements

We first examine how the gate placement affects the task-loss feedback received during sparsification. We show that BN strongly suppresses the task sensitivity of filter-space gates, whereas BN-space gates remain directly coupled to the task objective.

Theorem 1 (Relative sensitivity to gate scaling)**.** *Let z_c denote the pre-BN activation of channel c. We consider training-mode BN, where the batch mean μ_c and variance σ_c^2 are recomputed after each change to the gate. Let β_c denote the BN bias. For D-Gating, a gate g_c is applied before BN, giving*

$$y_c^{\mathrm{D}}(g_c) = \alpha_c \frac{g_c(z_c - \mu_c)}{\sqrt{g_c^2\sigma_c^2 + \epsilon}} + \beta_c,$$

where α_c is the BN scale. For BN-Gating, the effective BN scale γ_c is directly reparameterized, giving

$$y_c^{\mathrm{BN}}(\gamma_c) = \gamma_c \frac{z_c - \mu_c}{\sqrt{\sigma_c^2 + \epsilon}} + \beta_c.$$

Let $\tilde{y}_c = y_c - \beta_c$ denote the input-dependent part of the channel output. The relative sensitivities to the corresponding gates are

$$\textit{D-Gating:} \quad \frac{d\tilde{y}_c/\tilde{y}_c}{dg_c/g_c} = \frac{\epsilon}{g_c^2\sigma_c^2 + \epsilon}, \qquad \textit{BN-Gating:} \quad \frac{d\tilde{y}_c/\tilde{y}_c}{d\gamma_c/\gamma_c} = 1.$$

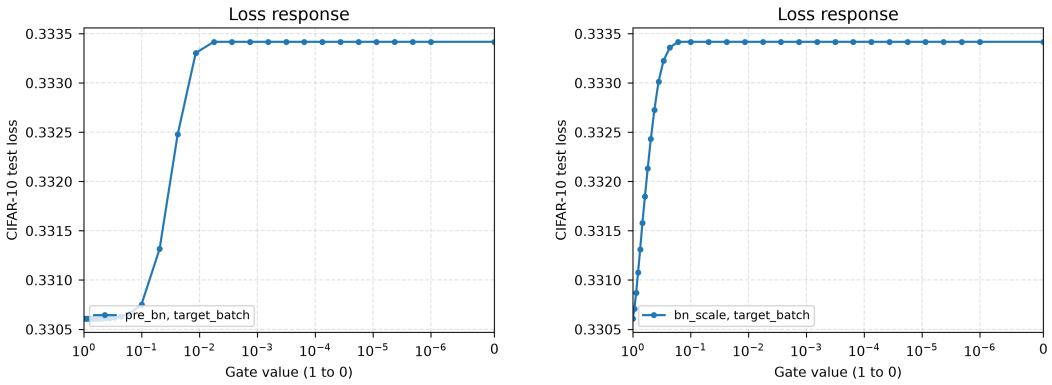
As shown in Theorem 1, when $g_c^2\sigma_c^2 \gg \epsilon$, the sensitivity of the normalized output to g_c is strongly suppressed, whereas its sensitivity to γ_c remains independent of the magnitude of γ_c.

This difference follows from where the gate is placed. In BN-Gating, changing γ_c directly changes the normalized channel output, so the task loss can immediately respond to sparsification. In D-Gating, decreasing g_c scales both the activation ($z_c \to g_c z_c$) and its BN standard deviation ($\sigma_c \to g_c\sigma_c$), causing the two effects to largely cancel. The task loss therefore provides weak feedback to g_c until $|g_c|$ approaches $\sqrt{\epsilon}/\sigma_c$, where this cancellation begins to break down.

Experimental evidence from varying the gate. To examine this difference, we vary the gate of the same channel in a pretrained model from 1 toward 0, placing it either before BN or on the BN scale. All other parameters remain fixed, while the target BN recomputes its statistics from the same input batch at each gate value. The resulting loss curves are shown in Figure 1.

Figure 1 shows that BN-Gating responds continuously as the gate decreases, whereas D-Gating remains nearly unchanged until the gate becomes very small. This agrees with the different gate sensitivities predicted by Theorem 1.

Experimental evidence from varying ϵ. To test whether BN normalization makes filter-space sparsification sensitive to ϵ, we vary ϵ within each method while keeping its other training settings fixed. Table 3 compares the resulting sparsity and test accuracy. As ϵ decreases from 10^{-3} to 10^{-7}, D-Gating sparsity falls substantially on both datasets, while BN-Gating sparsity remains basically unchanged. This contrast is consistent with the different gate sensitivities predicted above.



(a) D-Gating (b) BN-Gating

Figure 1: Test-loss responses as the same channel gate decreases from 1 to 0. The pre-BN gate leaves the loss nearly unchanged until it becomes very small, whereas the BN-scale gate changes the loss over a broader range.

Table 3: Effect of the BN parameter ϵ on D-Gating and BN-Gating.

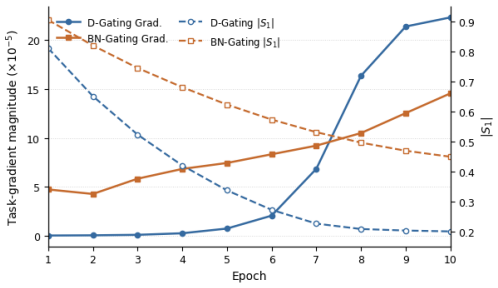
Dataset	ϵ	D-Gating		BN-Gating	
		Sparsity (%)	Acc. (%)	Sparsity (%)	Acc. (%)
CIFAR-10	10^{-3}	88.01 ± 0.78	90.94 ± 0.51	86.83 ± 0.11	92.26 ± 0.28
	10^{-5}	78.65 ± 1.57	91.76 ± 0.31	86.58 ± 0.18	92.22 ± 0.06
	10^{-7}	75.14 ± 1.87	91.96 ± 0.01	86.36 ± 0.07	92.28 ± 0.25
CIFAR-100	10^{-3}	73.53 ± 1.99	70.60 ± 1.21	73.94 ± 0.69	72.17 ± 0.40
	10^{-5}	58.96 ± 2.82	73.20 ± 0.17	73.95 ± 0.43	71.74 ± 0.35
	10^{-7}	30.70 ± 3.00	75.19 ± 0.20	73.92 ± 0.37	71.90 ± 0.43

4.2 Sparsity Patterns Induced by Task-Loss Feedback

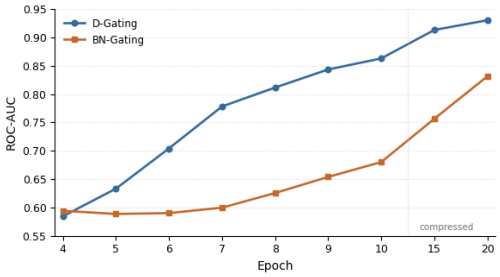
We next study how this difference in task-loss feedback translates into distinct sparsification dynamics over the course of training. Specifically, we compare how gate magnitudes and task gradients evolve in the two methods, and examine when gate values begin to reveal which channels will ultimately be pruned.

Task-gradient strength and gate magnitude. We track S_1, the channel-wise scalar factor multiplying the convolutional filter in D-Gating or forming the BN scale in BN-Gating. As shown in Figure 2a, $|S_1|$ falls rapidly in D-Gating but decreases more gradually in BN-Gating. The task-loss gradient received by S_1 is initially weak in D-Gating and becomes larger only after the gate has shrunk. This is consistent with BN largely absorbing filter scaling when the gate is large, allowing the sparsity regularizer to dominate its early updates. In BN-Gating, the gate receives task feedback throughout training, so the task loss and sparsity regularizer shape its magnitude together.

Channel-selection dynamics. To assess when the final sparse structure becomes predictable, we compute ROC-AUC using final pruning decisions as labels and the current $-|S_1|$ as scores. As shown in Figure 2b, the ROC-AUC for D-Gating rises sharply after its gates reach the scale-responsive regime, indicating that its channel selection becomes predictable within a relatively short stage after entering the regime. In contrast, BN-Gating's AUC increases gradually, suggesting that its sparse structure forms throughout training. These trends agree with the gradient analysis: D-Gating has a delayed, concentrated selection process, whereas task feedback shapes BN-Gating's channel selection over a longer period during training.

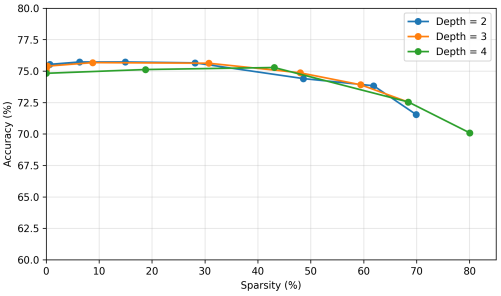


(a) Gate magnitude and task-gradient strength.

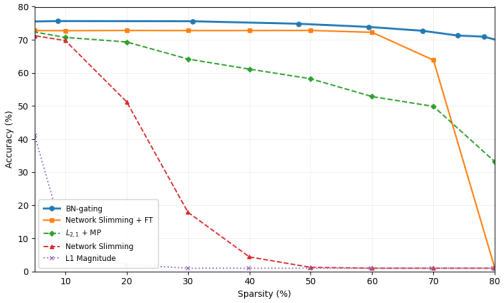


(b) Predictability of final channel sparsity.

Figure 2: Training dynamics of D-Gating and BN-Gating: gate magnitude and task-gradient strength (left), and predictability of final channel sparsity (right). D-Gating receives stronger task feedback only after its gates shrink, while BN-Gating receives task feedback throughout training and forms its sparse structure more gradually.



(a) BN-Gating with different gating depths on ResNet-18.



(b) Comparison with representative group-sparsity methods.

Figure 3: Sparsity–accuracy tradeoffs on CIFAR-100. BN-Gating achieves similar accuracy across gating depths (left) and retains higher accuracy than the compared pruning methods at high sparsity (right).

5 Function of Reparameterization for Sparsity Induction

This section studies how multiplicative reparameterization affects sparsity induction in BN space. We first examine the effect of factorization depth, then analyze the channel polarization induced by multiplicative reparameterization, and finally compare BN-Gating with other group sparsity-inducing methods.

5.1 Ablation on Gating Depth

We first study how the depth of multiplicative reparameterization affects sparsity induction in BN space. By varying the gating depth, we examine its influence on both the sparsity–accuracy tradeoff and the sensitivity of sparsity to the regularization strength, as shown in Figure 3a and Appendix Table 5, respectively.

Figure 3a shows that different gating depths achieve broadly similar sparsity–accuracy tradeoffs, although smaller depths tend to degrade more at very high sparsity. The main difference lies in how sparsity responds to the regularization strength. A larger gating depth is more aggressive, producing higher sparsity under the same regularization strength and larger changes in sparsity as the regularization is varied. In contrast, a smaller depth is more conservative and may require a much larger increase in regularization strength to further increase sparsity, especially in the high-sparsity regime. This makes the target sparsity harder to adjust efficiently. Depth 3 therefore provides a moderate sparsification rate while retaining sufficient control over the resulting sparsity level.

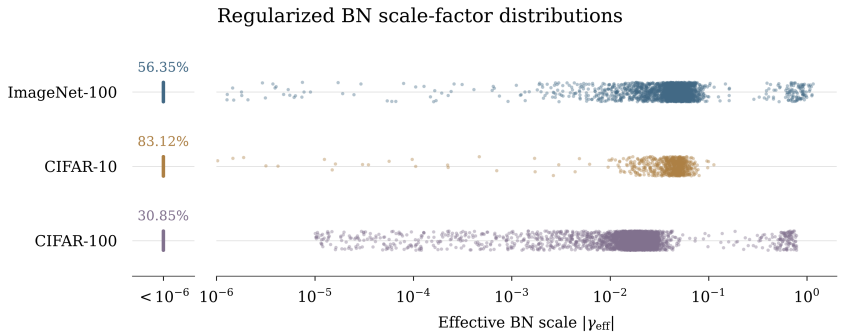


Figure 4: Distribution of effective BN scaling factors on CIFAR-10, CIFAR-100, and ImageNet-100. The separation between near-zero and larger scales illustrates the channel polarization induced by BN-Gating.

5.2 Multiplicative Reparameterization Induces Channel Polarization

A key benefit of multiplicative reparameterization is its *rich-get-richer* dynamic. For a channel parameterized as $s_c = \prod_{i=1}^{D} S_{c,i}$, the gradient of each factor is scaled by the product of the remaining factors. As a result, smaller channels receive weaker task gradients and are more easily pushed toward zero, while larger channels receive stronger task feedback and are more likely to remain active.

This feedback naturally amplifies differences between channels and leads to a polarized distribution, with one group concentrated near zero and the other clearly separated from zero. Such polarization makes the pruning threshold less ambiguous and reduces the dependence on post-pruning fine-tuning. Figure 4 visualizes the resulting separation.

5.3 Empirical Comparison with Other Group Sparsity-Inducing Methods

We further compare BN-Gating with several representative group sparsity-inducing methods on CIFAR-100, including Network Slimming, magnitude pruning, and $L_{2,1}$ regularization followed by magnitude pruning. We evaluate their sparsity–accuracy tradeoffs over a broad range of structured sparsity levels, as shown in Figure 3b.

In this comparison, BN-Gating maintains high accuracy over a substantially wider sparsity range. Network Slimming remains competitive at moderate sparsity after fine-tuning, but its performance decreases sharply in the high-sparsity regime. Magnitude-based methods degrade earlier, although $L_{2,1}$ regularization improves their robustness. In contrast, BN-Gating preserves more than 70% accuracy up to approximately 78% sparsity and remains above 67% accuracy at about 85% sparsity, demonstrating a more favorable sparsity–accuracy tradeoff under aggressive pruning.

6 Conclusion

We investigate how the placement of multiplicative reparameterization affects structured sparsification in networks with Batch Normalization and propose BN-Gating, which applies reparameterization to BN scaling factors. Across CIFAR-10, CIFAR-100, and ImageNet-100, BN-Gating achieves favorable sparsity–accuracy tradeoffs and identifies high-quality sparse architectures. Our analysis shows that BN-space gates remain directly coupled to the task loss, allowing task feedback to guide channel selection throughout training. Controlled gate-response and ϵ experiments further support this mechanism and reveal distinct sparsification dynamics. We also find that multiplicative reparameterization promotes polarized channel scales, with factorization depth controlling the strength of sparsity induction. Overall, these results establish BN-Gating as an effective approach to structured sparsification in networks with Batch Normalization.

REFERENCES

Zhen Dai, Mina Karzand, and Nathan Srebro. Representation costs of linear neural networks: Analysis and design. In *Advances in Neural Information Processing Systems*, volume 34, 2021.

Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*, 2019.

Trevor Gale, Erich Elsen, and Sara Hooker. The state of sparsity in deep neural networks. *arXiv preprint arXiv:1902.09574*, 2019.

Ariel Gordon, Elad Eban, Ofir Nachum, Bo Chen, Hao Wu, Tien-Ju Yang, and Edward Choi. Morphnet: Fast & simple resource-constrained structure learning of deep networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1586–1595, 2018.

Song Han, Jeff Pool, John Tran, and William J. Dally. Learning both weights and connections for efficient neural network. In *Advances in Neural Information Processing Systems*, volume 28, 2015.

Song Han, Huizi Mao, and William J. Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. In *International Conference on Learning Representations*, 2016.

Yang He, Ping Liu, Ziwei Wang, Zhilan Hu, and Yi Yang. Filter pruning via geometric median for deep convolutional neural networks acceleration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4340–4349, 2019.

Yang He, Yuhang Ding, Ping Liu, Linchao Zhu, Hanwang Zhang, and Yi Yang. Learning filter pruning criteria for deep convolutional neural networks acceleration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2009–2018, 2020.

Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*, 2017.

Torsten Hoefler, Dan Alistarh, Tal Ben-Nun, Nikoli Dryden, and Alexandra Peste. Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks. *Journal of Machine Learning Research*, 22(241):1–124, 2021.

Peter D. Hoff. Lasso, fractional norm and structured sparse estimation using a hadamard product parametrization. *Computational Statistics & Data Analysis*, 115:186–198, 2017.

Zehao Huang and Naiyan Wang. Data-driven sparse structure selection for deep neural networks. In *European Conference on Computer Vision*, pp. 317–334, 2018.

Chris Kolb, Laetitia Frost, Bernd Bischl, and David Rügamer. Differentiable sparsity via d-gating: Simple and versatile structured penalization. In *Advances in Neural Information Processing Systems*, volume 38, 2025a.

Chris Kolb, Tobias Weber, Bernd Bischl, and David Rügamer. Deep weight factorization: Sparse learning through the lens of artificial symmetries. In *International Conference on Learning Representations*, 2025b.

Chris Kolb, Christian L. Müller, Bernd Bischl, and David Rügamer. Smoothing the edges: Smooth optimization for sparse regularization using hadamard overparametrization. *Machine Learning*, 115, 2026.

Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. In *International Conference on Learning Representations*, 2017.

Jiangyuan Li, Thanh V. Nguyen, Chinmay Hegde, and Raymond K. W. Wong. Implicit sparse regularization: The impact of depth and early stopping. In *Advances in Neural Information Processing Systems*, volume 34, 2021.

Jiangyuan Li, Thanh V. Nguyen, Chinmay Hegde, and Raymond K. W. Wong. Implicit regularization for group sparsity. In *International Conference on Learning Representations*, 2023.

Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. Learning efficient convolutional networks through network slimming. In *Proceedings of the IEEE International Conference on Computer Vision*, 2017.

Christos Louizos, Max Welling, and Diederik P. Kingma. Learning sparse neural networks through l_0 regularization. In *International Conference on Learning Representations*, 2018.

Jian-Hao Luo, Jianxin Wu, and Weiyao Lin. Thinet: A filter level pruning method for deep neural network compression. In *Proceedings of the IEEE International Conference on Computer Vision*, 2017.

Paul Michel, Omer Levy, and Graham Neubig. Are sixteen heads really better than one? In *Advances in Neural Information Processing Systems*, volume 32, pp. 14014–14024, 2019.

Dmitry Molchanov, Arsenii Ashukha, and Dmitry Vetrov. Variational dropout sparsifies deep neural networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 2498–2507. PMLR, 2017.

Pavlo Molchanov, Arun Mallya, Stephen Tyree, Iuri Frosio, and Jan Kautz. Importance estimation for neural network pruning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11264–11272, 2019.

Clarice Poon and Gabriel Peyré. Smooth bilevel programming for sparse regularization. In *Advances in Neural Information Processing Systems*, 2021.

Clarice Poon and Gabriel Peyré. Smooth over-parameterized solvers for non-smooth structured optimization. *Mathematical Programming*, 201:897–952, 2023.

Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Learning structured sparsity in deep neural networks. In *Advances in Neural Information Processing Systems*, 2016.

Huangyu Xu, Jingqin Yang, Qianqian Xu, and Jiaye Teng. Theoretical analysis of sparse optimization with reparameterization, weight decay, and adaptive learning rate. In *International Conference on Machine Learning*, 2026.

Jianbo Ye, Xin Lu, Zhe Lin, and James Z. Wang. Rethinking the smaller-norm-less-informative assumption in channel pruning of convolution layers. In *International Conference on Learning Representations*, 2018.

Zhonghui You, Kun Yan, Jinmian Ye, Meng Ma, and Ping Wang. Gate decorator: Global filter pruning method for accelerating deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Michael Zhu and Suyog Gupta. To prune, or not to prune: Exploring the efficacy of pruning for model compression. *arXiv preprint arXiv:1710.01878*, 2017.

Liu Ziyin and Zihao Wang. spred: Solving l1 penalty with sgd. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 43407–43422, 2023.

Table 4: Training configurations used in the main experiments. LR denotes the initial learning rate.

Dataset	Method	Backbone	Depth	Epochs	Batch size	LR	LR schedule	Nesterov
CIFAR-10	BN-Gating	VGG-16	4	200	256	0.1	Cosine	Yes
CIFAR-10	D-Gating	VGG-16	4	200	256	0.4	Cosine	Yes
CIFAR-100	BN-Gating	ResNet-18	3	200	256	0.1	Cosine	Yes
CIFAR-100	D-Gating	ResNet-18	3	200	256	0.3	Cosine	Yes
ImageNet-100	BN-Gating	ResNet-18	3	100	256	0.1	5-epoch warm-up + cosine	No
ImageNet-100	D-Gating	ResNet-18	4	100	256	0.1	5-epoch warm-up + cosine	No

A AI USAGE DISCLOSURE

Generative AI tools were used to polish the writing and improve the clarity and readability of the manuscript. The authors take full responsibility for the final content.

B COMPLETE EXPERIMENTAL SETUP

B.1 EXPERIMENTAL SETUP

We evaluate BN-Gating and D-Gating on CIFAR-10, CIFAR-100, and ImageNet-100. All models are trained from scratch using cross-entropy loss with label smoothing of 0.1. We use SGD with momentum 0.9 and weight decay 10^{-4} for ordinary model parameters. Each experiment is repeated using three random seeds, $\{123, 124, 125\}$, and the results are reported as mean $\pm$ standard deviation. The main training configurations are summarized in Table 4.

For CIFAR-10, we use VGG-16 and apply random horizontal flipping, translations of up to 12.5%, and random rotations within $\pm 15^\circ$. For CIFAR-100, we use the CIFAR variant of ResNet-18, in which the first layer is replaced by a 3×3 stride-one convolution and the initial max-pooling layer is removed. The training augmentation consists of random horizontal flipping, random cropping with four-pixel reflection padding, and random rotations within $\pm 15^\circ$. For both CIFAR datasets, 5% of the original training set is held out for validation.

For ImageNet-100, we use the standard ResNet-18 architecture. Training images are processed using random resized cropping to 224×224 and random horizontal flipping. Validation images are resized to 256 pixels and center-cropped to 224×224.

For the CIFAR experiments, we follow the training configurations reported in *Differentiable Sparsity via D-Gating: Simple and Versatile Structured Penalization*. Specifically, we use an initial learning rate of 0.4 for four-factor D-Gating on CIFAR-10 and 0.3 for three-factor D-Gating on CIFAR-100, since D-Gating benefits from relatively large learning rates.

ImageNet-100 was not evaluated in the D-Gating paper. We therefore select $D = 4$ for ImageNet-100 because four-factor D-Gating provides the strongest overall sparsity–accuracy tradeoff across the experiments reported in that paper. For this experiment, we use an initial learning rate of 0.1 together with the standard ImageNet training schedule consisting of a five-epoch linear warm-up followed by cosine decay.

After training, the multiplicative factors are folded into the corresponding BN scaling factors or convolutional filters. The resulting sparse models are evaluated directly without post-pruning fine-tuning.

Architecture retraining. The compact architectures selected by BN-Gating and D-Gating are retrained from random initialization for 200 epochs. Both methods use a batch size of 256, an initial learning rate of 0.1 with cosine decay, and Nesterov SGD with momentum 0.9. Weight decay is set to 10^{-4} and label smoothing to 0.1. The channel pruning ratio is 73.83% on CIFAR-100 and approximately 83% on CIFAR-10. Data preprocessing and augmentation follow the corresponding main experiments.

Table 5: BN-Gating channel sparsity (%) at each regularization strength on CIFAR-100 with ResNet-18. Dashes indicate settings not evaluated.

λ	$D=2$	$D=3$	$D=4$
0.0005	–	0.00	0.00
0.0007	–	0.24	–
0.0010	0.67	8.80	18.75
0.0015	6.29	30.74	43.08
0.0020	14.96	47.98	–
0.0025	28.10	59.42	68.44
0.0030	–	68.19	–
0.0035	48.60	–	80.02
0.0045	61.88	–	–
0.0055	69.92	–	–

BN epsilon ablation. We evaluate both D-Gating and BN-Gating at $\epsilon \in \{10^{-3}, 10^{-5}, 10^{-7}\}$ on CIFAR-10 and CIFAR-100. For D-Gating, we use $\lambda = 0.0015$, $D = 4$, and an initial learning rate of 0.4 on CIFAR-10; on CIFAR-100, we use $\lambda = 0.0035$, $D = 3$, and an initial learning rate of 0.3. For BN-Gating, we use $\lambda = 0.0025$, $D = 4$, and an initial learning rate of 0.1 on CIFAR-10; on CIFAR-100, we use $\lambda = 0.0035$, $D = 3$, and an initial learning rate of 0.1. All models are trained for 200 epochs with a batch size of 256. Within each method and dataset, only ϵ is varied; all other settings follow the corresponding main experiments.

Factorization-depth ablation. We compare BN-Gating with $D \in \{2, 3, 4\}$ on CIFAR-100 using ResNet-18. All configurations use 200 training epochs, a batch size of 256, and an initial learning rate of 0.1 with cosine decay. For each depth, we vary λ to obtain different sparsity levels and apply weight decay λ/D to each factor. The remaining settings are shared with the main BN-Gating experiment.

Gate-response and gradient diagnostics. The gate-response experiment uses a pretrained VGG-16 with BN on CIFAR-10. We vary a scalar gate on the same channel either before BN or on the BN scale, using gate values from 1 down to 10^{-6} and an additional zero-gate evaluation. For the training-dynamics analysis, we use ResNet-18 on CIFAR-100 with $D = 3$ and track the first explicit scalar gate and its task-loss gradient. Measurements are recorded every epoch for the first 15 epochs and every five epochs thereafter.

Pruning baselines. The ℓ_1-magnitude pruning and Network Slimming baselines use ResNet-18 on CIFAR-100 and are trained for 200 epochs with a batch size of 256. Both use Nesterov SGD with momentum 0.9 and an initial learning rate of 0.2 with cosine decay. For $\ell_{2,1}$+MP, we use the same training duration and learning-rate schedule with a batch size of 128, and sweep $\lambda \in \{0, 10^{-4}, 2 \times 10^{-4}, 5 \times 10^{-4}, 7 \times 10^{-4}, 10^{-3}, 2 \times 10^{-3}, 3 \times 10^{-3}, 4 \times 10^{-3}\}$. All baselines use label smoothing of 0.1 and the same CIFAR-100 preprocessing and augmentation as the main experiments.

C Theoretical FLOPs of BN-Gating

We estimate ideal FLOPs from the channel masks produced by BN-Gating. After removing pruned channels, we count the convolutional and linear operations using the remaining layer widths, with one multiply–accumulate operation counted as two FLOPs.

D Pruning Interpretation

For BN-Gating, setting $\gamma_c = 0$ gives the constant BN output $y_c = \beta_c$. For D-Gating, setting the effective filter $W_c = 0$ makes its pre-BN activation zero, so its BN output is also constant under fixed inference statistics. In either case, the convolution producing that channel can be skipped in an equivalent implementation of the thresholded model, provided the constant contribution is retained. For a ResNet block, the same reasoning applies within the residual branch; the shortcut remains

Table 6: Idealized computation of BN-Gating channel configurations. Values are mean ± standard deviation over three seeds.

Dataset	Channel sparsity (%)	GFLOPs
CIFAR-10	77.23 ± 0.46	0.2055 ± 0.0143
	83.35 ± 0.30	0.1285 ± 0.0075
	86.94 ± 0.22	0.0889 ± 0.0063
CIFAR-100	30.74 ± 0.32	0.6718 ± 0.0019
	47.98 ± 0.50	0.4141 ± 0.0153
	68.19 ± 0.27	0.1770 ± 0.0108
	78.20 ± 0.10	0.0925 ± 0.0077
ImageNet-100	55.64 ± 0.73	1.2913 ± 0.0084

active. If the entire residual branch has constant outputs, its convolutions can be skipped and the constant branch output added to the shortcut.

E Proof of Theorem 1

Consider channel c and let $z_{c,i}$ denote its pre-BN activation at batch–spatial position i. We denote its batch mean and variance by

$$\mu_c = \mathbb{E}_i[z_{c,i}], \qquad \sigma_c^2 = \mathbb{E}_i[(z_{c,i} - \mu_c)^2]. \tag{3}$$

For a nonzero gate a, define the relative sensitivity of an input-dependent output $\widetilde{y}$ as

$$\mathcal{S}_a(\widetilde{y}) = \frac{a}{\widetilde{y}} \frac{\partial \widetilde{y}}{\partial a}. \tag{4}$$

Filter-space gating. Let g_c multiply the convolutional output before BN. Its mean and variance after gating are

$$\mu_c(g_c) = g_c \mu_c, \qquad \sigma_c^2(g_c) = g_c^2 \sigma_c^2. \tag{5}$$

Let α_c and β_c be the affine BN parameters. The input-dependent part of the BN output is

$$\widetilde{y}_{c,i}(g_c) = y_{c,i}(g_c) - \beta_c = \alpha_c \frac{g_c(z_{c,i} - \mu_c)}{\sqrt{g_c^2 \sigma_c^2 + \epsilon}}. \tag{6}$$

Differentiating with respect to g_c gives

$$\frac{\partial \widetilde{y}_{c,i}}{\partial g_c} = \alpha_c (z_{c,i} - \mu_c) \left[\frac{1}{\sqrt{g_c^2 \sigma_c^2 + \epsilon}} - \frac{g_c^2 \sigma_c^2}{(g_c^2 \sigma_c^2 + \epsilon)^{3/2}} \right] \tag{7}$$

$$= \alpha_c (z_{c,i} - \mu_c) \frac{\epsilon}{(g_c^2 \sigma_c^2 + \epsilon)^{3/2}}. \tag{8}$$

Therefore,

$$\mathcal{S}_{g_c}(\widetilde{y}_{c,i}) = \frac{g_c}{\widetilde{y}_{c,i}} \frac{\partial \widetilde{y}_{c,i}}{\partial g_c} = \frac{\epsilon}{g_c^2 \sigma_c^2 + \epsilon}. \tag{9}$$

Consequently, when $g_c^2 \sigma_c^2 \gg \epsilon$, the relative effect of changing the filter-space gate is strongly suppressed by BN.

BN-space gating. For BN-Gating, the effective scale γ_c is applied after normalization:

$$\widetilde{y}_{c,i}(\gamma_c) = \gamma_c \widehat{z}_{c,i}, \qquad \widehat{z}_{c,i} = \frac{z_{c,i} - \mu_c}{\sqrt{\sigma_c^2 + \epsilon}}. \tag{10}$$

Since $\widehat{z}_{c,i}$ does not depend on γ_c,

$$\frac{\partial \widetilde{y}_{c,i}}{\partial \gamma_c} = \widehat{z}_{c,i}, \tag{11}$$

and hence

$$\mathcal{S}_{\gamma_c}(\widetilde{y}_{c,i}) = \frac{\gamma_c}{\widetilde{y}_{c,i}} \frac{\partial \widetilde{y}_{c,i}}{\partial \gamma_c} = 1. \tag{12}$$

This proves the claimed difference in relative sensitivity.

Implication for task-loss feedback. Let $\delta_{c,i} = \partial\mathcal{L}/\partial\widetilde{y}_{c,i}$. The gradients with respect to the logarithmic gate magnitudes satisfy

$$\frac{\partial\mathcal{L}}{\partial\log|g_c|} = \frac{\epsilon}{g_c^2\sigma_c^2+\epsilon}\sum_i \delta_{c,i}\widetilde{y}_{c,i}, \tag{13}$$

$$\frac{\partial\mathcal{L}}{\partial\log|\gamma_c|} = \sum_i \delta_{c,i}\widetilde{y}_{c,i}. \tag{14}$$

Thus, the task-gradient component acting on a filter-space gate is attenuated by the same BN-dependent factor, whereas a BN-space gate remains directly coupled to the task loss throughout sparsification.

F Implicit Regularization of D-Gating and BN-Gating

Although both methods apply quadratic regularization to their multiplicative factors, they induce different effective sparse regularizers. D-Gating induces a group penalty on convolutional filters, whereas BN-Gating induces an element-wise penalty on BN scaling factors. We derive the two results separately below.

For clarity, we omit the conventional factor $1/2$ in squared ℓ_2 penalties. Including it only rescales λ and does not change the induced exponent.

F.1 D-Gating Induces Group $L_{2,2/D}$ Regularization

Let $w_c \in \mathbb{R}^{m_c}$ denote the convolutional filter associated with output channel c. A depth-D D-Gating parameterization represents this filter as

$$w_c = A_c \prod_{d=2}^{D} S_{d,c}, \tag{15}$$

where A_c is the base filter and $S_{2,c},\dots,S_{D,c}$ are channel-wise scalar factors. The quadratic regularization cost of this representation is

$$R_D(A_c,S_c) = \frac{1}{D}\left(\|A_c\|_2^2 + \sum_{d=2}^{D} S_{d,c}^2\right). \tag{16}$$

We define the representation cost of an effective filter as the minimum quadratic penalty over all its equivalent factorizations:

$$\mathcal{R}_D(w_c) = \min_{A_c,\{S_{d,c}\}_{d=2}^{D}} R_D(A_c,S_c) \quad \text{s.t.} \quad w_c = A_c\prod_{d=2}^{D} S_{d,c}. \tag{17}$$

Proposition 1. The minimum representation cost of D-Gating is

$$\mathcal{R}_D(w_c) = \|w_c\|_2^{2/D}. \tag{18}$$

Proof. Let

$$r_1 = \|A_c\|_2, \qquad r_d = |S_{d,c}|, \qquad d = 2,\dots,D. \tag{19}$$

The factorization implies

$$\|w_c\|_2 = \prod_{d=1}^{D} r_d. \tag{20}$$

Applying the arithmetic–geometric mean inequality to $r_1^2,\dots,r_D^2$ gives

$$R_D(A_c,S_c) = \frac{1}{D}\sum_{d=1}^{D} r_d^2 \tag{21}$$

$$\geq \left(\prod_{d=1}^{D} r_d^2\right)^{1/D} \tag{22}$$

$$= \|w_c\|_2^{2/D}. \tag{23}$$

It remains to show that the lower bound is attainable. For $w_c \neq 0$, choose

$$A_c = \frac{w_c}{\|w_c\|_2^{(D-1)/D}}, \qquad S_{d,c} = \|w_c\|_2^{1/D}, \qquad d = 2, \ldots, D. \tag{24}$$

This construction satisfies

$$\|A_c\|_2 = |S_{2,c}| = \cdots = |S_{D,c}| = \|w_c\|_2^{1/D}, \tag{25}$$

and reconstructs w_c exactly. Therefore, equality holds in the bound above. The result is immediate for $w_c = 0$ by setting all factors to zero.

Summing the minimum representation cost over all convolutional filters gives

$$\sum_c \mathcal{R}_D(w_c) = \sum_c \|w_c\|_2^{2/D} = \|w\|_{2,2/D}^{2/D}. \tag{26}$$

Consequently, the factorized D-Gating objective

$$\mathcal{L}(w) + \frac{\lambda}{D} \sum_c \left(\|A_c\|_2^2 + \sum_{d=2}^{D} S_{d,c}^2 \right) \tag{27}$$

induces the effective objective

$$\mathcal{L}(w) + \lambda \|w\|_{2,2/D}^{2/D}. \tag{28}$$

This is precisely a group-sparsity penalty: the inner ℓ_2 norm measures the magnitude of each filter, while the outer exponent $2/D$ promotes sparsity across filters.

F.2 BN-Gating Induces $\ell_{2/D}$ Regularization

BN-Gating applies multiplicative reparameterization to the scalar BN scale instead of the convolutional filter. For channel c, let

$$\gamma_c = \prod_{d=1}^{D} S_{d,c}, \tag{29}$$

where $S_{1,c}, \ldots, S_{D,c}$ are scalar factors. Their quadratic regularization cost is

$$R_{\mathrm{BN}}(S_c) = \frac{1}{D} \sum_{d=1}^{D} S_{d,c}^2. \tag{30}$$

The minimum cost of representing an effective BN scale is

$$\mathcal{R}_{\mathrm{BN}}(\gamma_c) = \min_{\{S_{d,c}\}_{d=1}^{D}} R_{\mathrm{BN}}(S_c) \quad \text{s.t.} \quad \gamma_c = \prod_{d=1}^{D} S_{d,c}. \tag{31}$$

Proposition 2. The minimum representation cost of BN-Gating is

$$\mathcal{R}_{\mathrm{BN}}(\gamma_c) = |\gamma_c|^{2/D}. \tag{32}$$

Proof. Applying the arithmetic–geometric mean inequality to the squared factors gives

$$R_{\mathrm{BN}}(S_c) = \frac{1}{D} \sum_{d=1}^{D} S_{d,c}^2 \tag{33}$$

$$\geq \left(\prod_{d=1}^{D} S_{d,c}^2 \right)^{1/D} \tag{34}$$

$$= |\gamma_c|^{2/D}. \tag{35}$$

For $\gamma_c \neq 0$, the lower bound is attained by the balanced factorization

$$S_{1,c} = \mathrm{sign}(\gamma_c)|\gamma_c|^{1/D}, \qquad S_{d,c} = |\gamma_c|^{1/D}, \qquad d = 2, \ldots, D. \tag{36}$$

All factors then have the same magnitude and their product is exactly γ_c. For $\gamma_c = 0$, setting all factors to zero attains zero cost. Therefore, $\mathcal{R}_{\mathrm{BN}}(\gamma_c) = |\gamma_c|^{2/D}$.

Summing over all BN channels yields

$$\sum_c \mathcal{R}_{\mathrm{BN}}(\gamma_c) = \sum_c |\gamma_c|^{2/D} = \|\gamma\|_{2/D}^{2/D}. \tag{37}$$

Therefore, the factorized BN-Gating objective

$$\mathcal{L}(\gamma) + \frac{\lambda}{D} \sum_c \sum_{d=1}^{D} S_{d,c}^2 \tag{38}$$

induces the effective objective

$$\mathcal{L}(\gamma) + \lambda \|\gamma\|_{2/D}^{2/D}. \tag{39}$$

F.3 Comparison

Both methods induce the same exponent

$$p = \frac{2}{D}, \tag{40}$$

but apply it to different effective quantities:

$$\text{D-Gating:} \quad \sum_c \|w_c\|_2^p, \tag{41}$$

$$\text{BN-Gating:} \quad \sum_c |\gamma_c|^p. \tag{42}$$

D-Gating therefore induces group sparsity through the norms of convolutional filters, whereas BN-Gating directly induces sparsity in the scaling factors of normalized channels.

The exponent becomes smaller as the factorization depth increases:

$$D = 2 \Rightarrow p = 1, \qquad D = 3 \Rightarrow p = \frac{2}{3}, \qquad D = 4 \Rightarrow p = \frac{1}{2}. \tag{43}$$

For $D = 2$, the induced penalties reduce to group Lasso for D-Gating and ℓ_1 regularization for BN-Gating. For $D > 2$, the induced penalties are nonconvex and more strongly favor sparse solutions. This explains why larger factorization depths generally induce sparsity more aggressively.

The equalities above are attained by balanced factors. If a factorization is unbalanced, it can be rescaled without changing the effective filter or BN scale while strictly decreasing its quadratic penalty. Consequently, any minimum of the factorized objective must admit a balanced representation. Multiplicative reparameterization thus provides a smooth optimization parameterization whose minimum representation cost is an explicit sparsity-inducing $L_{2,2/D}$ or $\ell_{2/D}$ penalty.